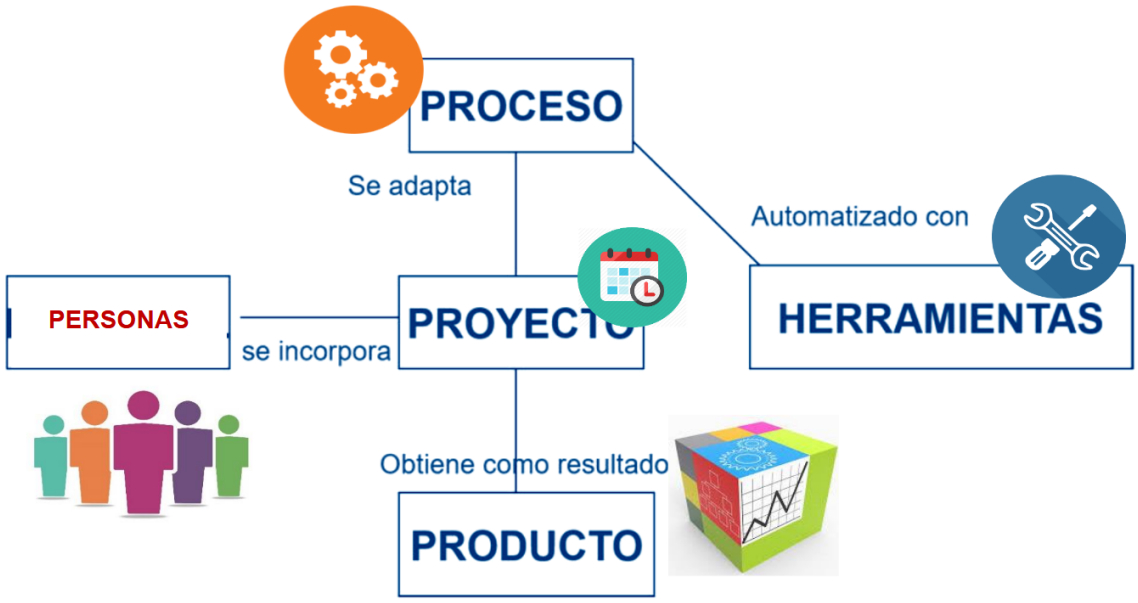
20/08/2025

Clase 02: SCM – Software Configuration Management

Es una disciplina que integra la Ingeniería de Software, nos sirve para mantener la integridad del software como producto ya que este es maleable, por lo que será necesario para afrontar los cambios que se presentarán.



- **Proceso**: Es una definición conceptual o teórica de lo que hay que hacer. Describe qué cosas se deben hacer, en qué orden, qué artefactos producir, y cómo se deben llevar a cabo las tareas.
- **Proyecto**: Es una **unidad de gestión** o de organización del trabajo. Es el medio que permite organizar a las personas y administrar los recursos para obtener un resultado. El proceso se instancia en un proyecto, esto significa que las tareas teóricas definidas en el proceso se ejecutan concretamente en el proyecto.
  - Tiene un resultado único, cada producto es distinto.
  - Tiene un esfuerzo temporal definido, o sea, una fecha de inicio y fecha de fin.
  - Es de elaboración gradual, el objetivo se descompone y se alcanza de a poco a través de tareas interrelacionadas.
- **Producto**: Es el resultado o la salida de un proyecto y del proceso de desarrollo de software.

¿Qué es el **Software**?

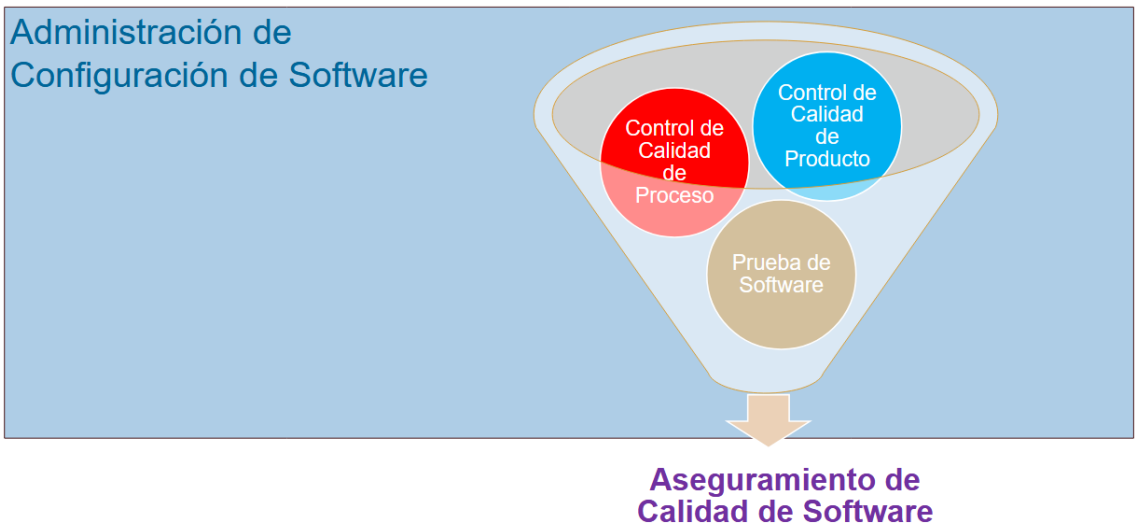
Consiste en un conjunto de programas, procedimientos, reglas, documentación y datos. Es información estructurada con propiedades lógicas y funcionales, creada y mantenida en varias formas y representaciones, diseñada para ser procesada por una computadora.

¿Qué produce los **Cambios en el Software**?

- Cambios del negocio y nuevos requerimientos.
- Soporte de cambios de productos asociados.

- Reorganización de las prioridades de la empresa por crecimiento.
- Cambios en el presupuesto.
- Defectos encontrados a corregir.
- Oportunidades de mejora.

Disciplinas de soporte del Software



No se debe confundir **Aseguramiento de Calidad de Software** con **Control de Calidad**, el aseguramiento busca la calidad antes de que el producto esté terminado. Estos permiten, mediante la definición de sus procesos, asegurar que los procesos que se ejecutan generen productos que sean de calidad.

¿Qué significa que un **producto sea íntegro**?

- Satisface las necesidades del usuario.
- Puede ser fácil y completamente rastreado durante su ciclo de vida.
- Satisface criterios de performance.
- Cumple con sus expectativas de costo.

Y, ¿Qué **problemas** se presentan cuando **manejo componentes**?

- Pérdida de un componente
- Pérdida de cambios (el componente que tengo no es el último)
- Sincronía fuente-objeto–ejecutable
- Regresión de fallas
- Doble mantenimiento
- Superposición de cambios
- Cambios no validados

Conceptos clave para SCM

- Ítem de Configuración
- Repositorio
- Línea Base
- Ramas (Branch)
- Configuración del Software

Ítem de Configuración

Son todos y cada uno de los artefactos que forman parte del producto o del proyecto, que pueden sufrir cambios o necesitan ser compartidos entre los miembros del equipo y sobre los cuales necesitamos conocer su estado y evolución.

¿**Qué puede ser** un Ítem de Configuración?

Es cualquier cosa que se pueda meter en un filesystem. Hay un listado, este es:

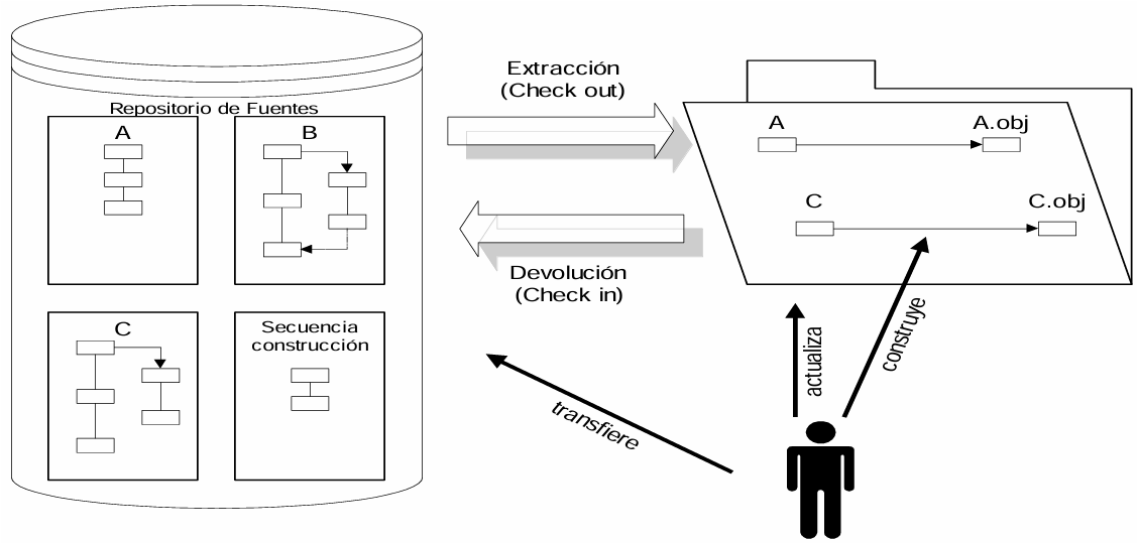
- Plan de CM
- Propuestas de Cambio
- Visión
- Riesgos
- Plan de desarrollo
- Prototipo de Interfaz
- Guía de Estilo de IHM
- Manual de Usuario
- Requerimientos
- Plan de Calidad
- Arquitectura del Software
- Plan de Integración
- Planes de Iteración
- Estándares de codificación
- Casos de prueba
- Código fuente
- Gráficos, iconos, ...
- Instructivo de ensamble
- Programa de instalación
- Documento de despliegue
- Lista de Control de entrega
- Formulario de aceptación
- Registro del proyecto

Versión y Variante

- Una versión se define, desde el punto de vista de la evolución, como la forma particular de un artefacto en un instante o contexto dado.
- Una variante es una versión de un ítem de configuración (o de la configuración) que evoluciona por separado.

Repositorio

Un repositorio es un contenedor de información que almacena los ítems de configuración (ICs). Puede ser un directorio, una base de datos (relacional o de objetos), o una combinación de ambas.



- **Check-out** (Extracción): Permite obtener la última versión (o versiones anteriores) de los ítems de configuración y traerlos a un directorio local para su modificación.
- **Check-in** (Devolución): Sirve para introducir las versiones modificadas de los ítems de configuración de vuelta al repositorio.

Tipos de repositorios:

- **Centralizados:** Un único servidor contiene todos los archivos con sus versiones. Los administradores tienen mayor control, pero una falla en el servidor puede ser crítica
    - Las acciones de check-in y check-out se limitan a la última versión o a lo que se va a cambiar.
- **Descentralizados:** Cada cliente tiene una copia idéntica del repositorio completo. Esto ofrece mayor resiliencia (si un servidor falla, se puede "copiar y pegar" desde un cliente) y posibilita otros flujos de trabajo.
    - Permiten traer una versión completa del repositorio, lo cual es más costoso en espacio, pero más cómodo.

Línea Base

Una línea base es una configuración del software que ha sido formalmente revisada y acordada, sirviendo como un punto seguro y una base para desarrollos posteriores. Se considera una versión íntegra de un conjunto de ítems de configuración en un momento determinado.

Ramas

Creación de ramas

- Existe una rama principal (trunk, master).
- Sirven para bifurcar el desarrollo.
- Pueden tener razones de creación con semántica.

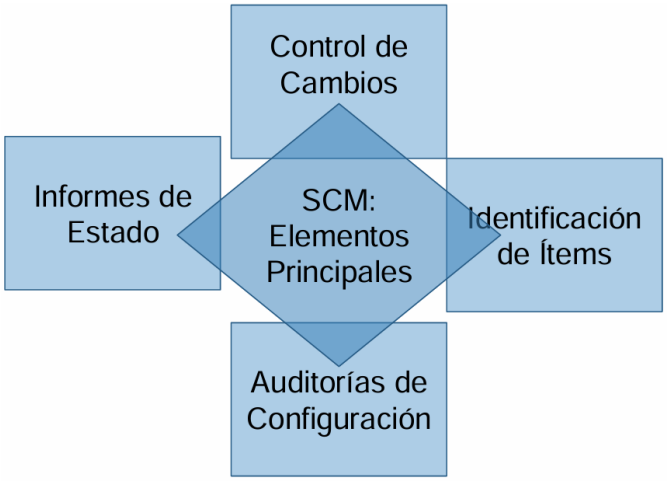
- Permiten la experimentación.
- Pueden ser descartadas o integradas.

Integración de ramas

- La operación se llama merge.
- Lleva los cambios a la rama principal.
- Pueden surgir conflictos (resolvemos con diff).
- Todas las ramas deberían eventualmente integrarse a la principal o ser descartadas.

Configuración del Software

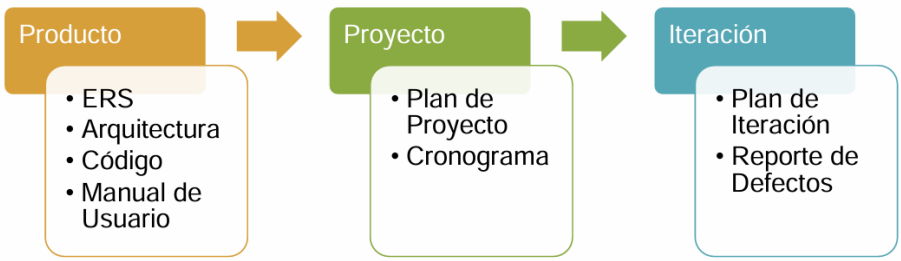
Actividades Fundamentales de la Gestión de Configuración del Software



Identificación de Ítems

Se define la estructura del repositorio, se identifican ítems unívocamente y se asignan a un lugar específico en la estructura del repositorio. Se definen también reglas de nombrado, convenciones, esquemas genéricos, etc.

Tipos de Ítems de Configuración



Los Ítems de Configuración de Iteración son de los que duran menos su ciclo de vida, ya que una vez que se termina la iteración estos dejan de ser tan relevantes.

Los relacionados al producto son los que más van a trascender a lo largo de los proyectos (un proyecto por cada iteración).

Control de Cambios

Un proceso de control de cambios es un proceso donde, a partir de una línea base, se pide hacer un control de cambios, para ello:

1. **Propuesta de Cambio** (o Petición de Cambios): Se genera una propuesta que detalla el cambio solicitado.
2. **Análisis de Impacto**: Se evalúan los aspectos técnicos, los efectos colaterales, los costos del proyecto y el impacto general que el cambio podría tener.
3. **Revisión de Partes**: La propuesta y el análisis de impacto son revisados por las partes interesadas.
4. **Decisión del Comité de Control de Cambios** (CCC): Este comité se encarga de analizar el cambio y su impacto para aceptar o rechazar la modificación.
5. **Notificación a Partes / Informar al Cliente**: Una vez tomada la decisión, se notifica a todas las partes involucradas, incluyendo al cliente.
6. **Generación de Orden de Cambios de Ingeniería**: En caso de aprobación, se genera una orden formal para implementar el cambio.
7. **Ubicación en la cola de cambios**: El cambio aprobado se coloca en una lista para su implementación.
8. **Verificación y Validación**: Una vez implementado el cambio, se verifica que se haya realizado correctamente y que el software resultante cumpla con los requerimientos esperados. Solo después de este paso, se podría establecer una nueva línea base.

Comité de Control de Cambios

- **Comité de Control de Cambios**: Gestiona cambios menores e incluye roles clave del equipo de desarrollo, como el analista que define la solución, el desarrollador que la implementará, el líder técnico y el líder de testing.
- **Comité de Control de Cambios Ampliado**: Interviene en cambios mayores que tienen un impacto significativo (económico, de producto a largo plazo, etc.). Este comité incluye figuras como el gerente de calidad o el responsable del producto, además de los líderes de proyecto.

Auditorías de Configuración

- **Auditoría Física de Configuración** (PCA - Physical Configuration Audit): Asegura que lo que está indicado para cada Ítem de Configuración (IC) en la línea base o en la actualización se ha alcanzado realmente. Se centra en la verificación de que el producto se construye de la manera correcta.
- **Auditoría Funcional de Configuración** (FCA - Functional Configuration Audit): Es una evaluación independiente de los productos de software, controlando que la funcionalidad y el rendimiento reales de cada ítem de configuración sean consistentes con la especificación de requerimientos. Abarca tanto la verificación

(el producto se construye correctamente) como la validación (el producto correcto resuelve los problemas del cliente).

Estas auditorías tienen 2 características:

- **Objetividad**: Para que el proceso sea objetivo, la persona que ejecuta la auditoría debe ser externa al proyecto sobre el que se realiza la auditoría. Puede ser alguien de la organización, pero que no forme parte del equipo del proyecto. A menudo, el gerente de configuración lidera estas auditorías o se contrata a una auditora externa.
- **Resultados**: La salida de un proceso de auditoría es un informe de auditoría que contiene un listado de hallazgos (fallas, cosas a corregir), oportunidades de mejora y observaciones. También puede identificar cosas que se hicieron correctamente.

Informes de Estado

Su función principal es mantener los registros de la evolución del sistema. Estos informes manejan una gran cantidad de información y, por lo tanto, suelen implementarse dentro de procesos automáticos. Incluyen reportes de rastreabilidad de todos los cambios realizados a las líneas base durante el ciclo de vida del producto.

Algunas preguntas que un Informe de Estado puede responder:

- ¿Cuáles el estado del ítem?
- ¿Un requerimiento de cambio ha sido aprobado o rechazado por el CCB?
- ¿Qué versión de ítem implementa un requerimiento de cambio aprobado (saber cuál es el componente que contiene la mejora)?
- ¿Cuál es la diferencia entre una versión y otra dada?

Preguntas sobre la aplicación en Agile, según Google NotebookLM:

¿Qué pasa con el Comité de Control de Cambios?

En los ambientes Ágiles, la existencia y el funcionamiento del Comité de Control de Cambios es "medio discutido". La filosofía Ágil promueve responder a los cambios en lugar de tratar de evitarlos, y valora la colaboración con el cliente más que las negociaciones contractuales formales. El SCM en Ágil debe servir al equipo de desarrollo y coordinar el desarrollo en lugar de controlar a los desarrolladores. El proceso tradicional de pedir aprobación formal a un comité para cada cambio se considera contrario a la agilidad. En su lugar, la interacción y el feedback continuo con el cliente son fundamentales; si al cliente le gusta un cambio, se aprueba mediante esa colaboración y no a través de un proceso burocrático, priorizando mostrar software funcionando.

¿Qué ítems de configuración podemos tener?

En el contexto de SCM en Ágil, si bien la definición de ICs se mantiene, hay un fuerte enfoque en la "Documentación Lean y Trazabilidad" y en "Eliminar el desperdicio - no

agregar nada más que valor". Esto implica que la elección de qué ítems de configuración se mantendrán y con qué nivel de detalle se hará, buscará la eficiencia y la utilidad directa para el equipo y el cliente, priorizando el software funcionando sobre una documentación excesiva. Sin embargo, elementos esenciales como el código fuente, los requerimientos (a menudo expresados como User Stories), y las pruebas seguirán siendo ítems de configuración críticos.

¿Qué pasa con las auditorías?

Aunque los materiales no afirman que las auditorías desaparecen por completo, el enfoque Ágil prioriza el "Feedback continuo y visible sobre calidad, estabilidad e integridad" y la automatización de tantos procesos como sea posible. La naturaleza iterativa y de inspección-adaptación de los procesos empíricos (base de Ágil) sugiere que los controles de calidad y la verificación de la integridad se integran de manera continua en el desarrollo, en lugar de depender de auditorías formales y periódicas realizadas por terceros. La idea de "no hacer cosas de más" y "no documentar de más" implica una simplificación de procesos que no añaden valor directo o que pueden ser abordados de manera más ágil.

¿Qué pasa con los reportes de estado?

El agilismo busca eliminar el desperdicio y promover la "Documentación Lean y Trazabilidad". En lugar de informes formales y extensos, se valora el feedback continuo y visible sobre calidad, estabilidad e integridad. La automatización juega un papel clave, permitiendo que las herramientas de gestión de configuración proporcionen información en tiempo real sobre el estado del software. La idea es generar rápidamente versiones nuevas del software que puedan ser mostradas al cliente y que "realmente anden", en lugar de depender de informes detallados. Esto significa que el enfoque cambia de la generación manual de informes a la obtención de información de estado de manera más dinámica y automatizada, integrada en el flujo de trabajo continuo del equipo.

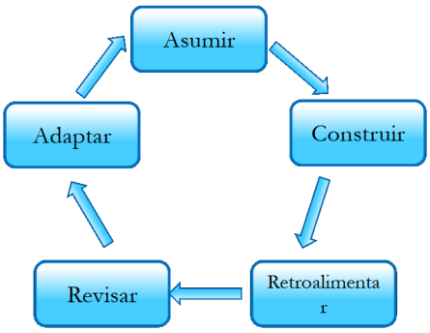
22/08/25

Clase 03: Filosofía Ágil y Manifiesto Ágil

Procesos empíricos

Los procesos empíricos son un tipo de enfoque para el desarrollo de software que se basa en la experiencia, el aprendizaje continuo y la adaptación constante, a diferencia de los procesos definidos que buscan la previsibilidad a través de pasos detallados y formales.

Patrón de conocimiento en los procesos empíricos



1. **Asumir**: Se parte de una hipótesis o de lo que se cree que es necesario hacer.
2. **Construir**: Se desarrolla el producto o se realiza el trabajo basado en esa asunción.
3. **Retroalimentar**: Se recibe el feedback o la realimentación sobre el trabajo que se ha construido. El feedback continuo y visible sobre la calidad, estabilidad e integridad del software es un aspecto clave de la inspección.
4. **Revisar**: Se procede a la revisión del trabajo realizado. Esto implica inspeccionar qué se hizo bien y qué se hizo mal, aprendiendo de la experiencia generada. La transparencia es crucial en esta etapa para que la inspección sea efectiva, implicando ser sincero y compartir información, sin ocultar problemas y pidiendo ayuda cuando sea necesario.
5. **Adaptar**: Como consecuencia de la inspección, se realizan los ajustes necesarios al proceso o al producto.

El manifiesto ágil

<u>Los 4 valores ágiles</u>

- Individuos e interacciones sobre procesos y herramientas: Esto significa que, aunque los procesos y las herramientas son valiosos, se prioriza a las personas y las relaciones interpersonales entre ellas.
- Software funcionando sobre documentación exhaustiva: Se valora más lo que entrega valor al cliente, es decir, el software que funciona, por encima de grandes cantidades de documentación detallada.
- Colaboración con el cliente sobre negociación contractual: Se prefiere una relación de colaboración integrada con el cliente, trabajando juntos, en lugar de aferrarse a contratos firmados y negociaciones formales.

- Respuesta al cambio sobre seguir un plan rígido: Se valora la capacidad de entender que habrá cambios y de incorporarlos, en lugar de aferrarse estrictamente a un plan predefinido.

Los 12 principios ágiles

¿Qué es **ágil**?

Es una ideología, **no** una metodología **ni** un proceso.

¿Qué significa **ser ágil**?

- Busca un **equilibrio entre la ausencia total de procesos y el exceso de formalidad y burocracia**.
- Los métodos ágiles son **adaptables** en lugar de predictivos, lo que significa que están diseñados para reaccionar y ajustarse a los cambios.
- Son **orientados a las personas** en lugar de orientados al proceso, priorizando las interacciones humanas.

¿Qué **métodos ágiles** hay disponibles?

- Scrum: Es un framework ágil muy conocido. Se menciona que tiene tres roles, tres categorías de responsabilidades, cuatro ceremonias o reuniones y tres artefactos. Es tan exigente que, si no se hace al pie de la letra de lo que dice, no se garantiza los resultados.
- XP (eXtreme Programming): Fue el primer framework que surgió en la industria. Se menciona que tiene 14 principios, y si no se cumplen, no se considera "programación extrema" y no se garantizan los resultados.

- Crystal: Este es un conjunto de prácticas que se especializan según el tipo de software que se va a construir. Por ejemplo, indica diferentes prácticas si el software es crítico (pone en riesgo vidas), solo afecta el confort, o involucra grandes cantidades de dinero.
- FDD (Feature Driven Development).
- ATDD (Acceptance Test Driven Development): Este enfoque lleva el Test Driven Development (desarrollo guiado por pruebas) hacia las pruebas de negocio, de modo que las prácticas de desarrollo están dirigidas a demostrar que los casos de negocio funcionan en el software.
- Shap: Este es un framework más reciente, mencionado como una propuesta que surge en "rebeldía" de Scrum.